Utilizing color data in VoteNet for 3D Object Detection in Point Clouds

Xue Dong
SCPD Student
dongxue {at} stanford {dot} edu

Abstract

RGB-D object detection has become a popular research topic due to the demand from application such as automated car and LIDAR data processing. Recent progress of deep learning techniques has greatly improved model precision and processing time. In this project, We study the state-of-the-art approaches of object detection techniques and compare them in terms of approaches they are using as well as performance. One of the recent approach VoteNet is studied in detail as baseline method. Based on the idea that RGB and depth information can be fused to improve performance of the method, We propose a method that uses 2D bounding box frustum class score as feature of point cloud. Evaluation is performed on SUN RGB v1 dataset.

1. Introduction

3D object detection aims at locating 3D object bounding boxes and classifying them. These techniques have been widely used in 3D graphics industry and recently automated driving. 3D cloud points scanned from depth sensor, or LIDAR, contain more information than 2D images from cameras.

Thanks to the recent development of deep learning concepts, 2D object detection accuracy has been greatly improved[10]. However due to the amount of data and computation in 3D world, object detection is more difficult than in 2D. On one hand if we use 2D CNN based methods in 3D, the computation of 3D convolution will be too heavy for current hardware. On another hand 3D cloud points are sparse samples, compared with 2D images which is a dense and nice stuctured sample in Euclidean space.

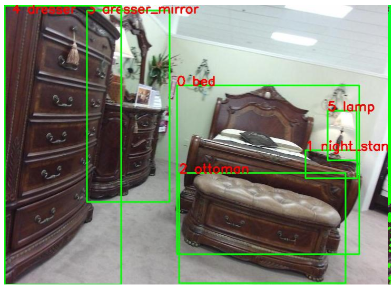
Motivation VoteNet[4] introduced a framework to detect objects in 3D point cloud using only depth information. This framework even out performed other approaches that use both depth and color information. Naturally, one would think the performance of VoteNet could be improved if the color information can be used. Moreover, combing color and depth information in object detection pipeline is an open topic. MV3D [1], AVOD [2] and more recently Frustum PointNets[5] are popular approaches. Moreover despite precision and resolution of 3D scanner or LiDAR have been greatly improved over recent years, 2D image quality is still considered better than 3D depth camera. In this project We will review those approaches and find a way to blend color information into VoteNet.

Contributions We am doing this project by myself. Before doing the project We was interested but not familiar with deep learning concept. By doing the background and literature review and code study of VoteNet, We will learn state-of-the-art approaches in object detection and deep learning frameworks, as well as popular ML open source libraries such as PyTorch and TensorFlow.

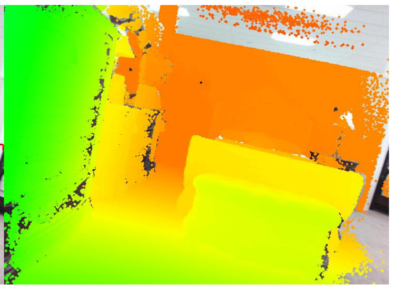
2. Related work

Point Cloud has geometry information that can be used for 3D object detection. Mehods using point cloud can be categorized into 3 groups:

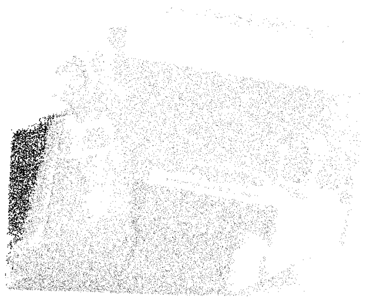
- Bird-Eye-View projects point cloud on to bird eye view image, and combine features from multiple channels. MV3D [1] introduced a framework that combines bird eye view, front view and RGB image to generate 3D bounding box proposal, AVOD [2] improved MV3D by using FPN framework to generate full resolution feature map, and improved performace on small objects by preserve details from multiple levels in the network.
- Voxel based methods transform point cloud into voxel grid, so that 3D convolution can be performed without huge amount of computation. Graham *et al.* introduced a sparse CNN method that accelerates computation by dropping some voxels that do not contain useful information.
- Original point cloud methods such as PointNet[6] and PointNet++[7] extract features from point clouds directly. Using MLP and max pooling, PointNet can generate features with global information. Further more, PointNet++ improves local feature extracting by grouping point cloud and running PointNet inside gourps.



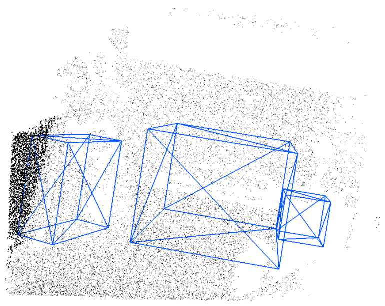
(a) Image of scene and ground truth



(b) Depth map(convert to location info)



(c) Input point cloud



(d) Bounding box result from VoteNet

Figure 1: Detection result of #1 scene from SUN RGB-D v1 dataset.

Hough Voting was first introduced to detec lines in 2D images. By transforming points in image into lines in Hough space, lines can be detected as cross points in Hough space. Further more, Generalized Hough Transform[3] extends this technique to detect any object in 2D images. The method first select features on an object as codebook, and then calculate offset of these features to the center of the object. At inference time, features in codebook are located on image, and then each match votes center of object by the offset learned from previos step. By clustering the votes, the center of an object can be found in an image.

2D image understanding task has many comparably mature techniques and algorithms. So when considering **3D object detection**, it is naturally to think about taking advantage of 2D object detectors. Qi *et al.* [5] achieve this by creating 2D bounding box on the RGB image, then lifting the area surrounded by the bounding box to a frustum. This frustum defines an area that points inside it very likely belongs to item that shown in the 2D bounding box.

3. RGB Data in VoteNet framework

3.1. Baseline method

The original VoteNet is introduced by Qi *et al.* [4]. Given a set of input 3D points $\{p_i\}_{i=1}^{N}$, the first step is to use PointNet++[7] to select seed points and enriched C-dimensional feature vector. Information of location of cloud points is embedded into $\{s_i\}_{i=1}^{M}$ where $s_i = [x_i; f_i]$ with $x_i \in \mathbb{R}^3$ and $f_i \in \mathbb{R}^C$. Then the seed points are given to a multi-layer perception network to compute vote $v_i = [y_i; g_i]$ where $y_i = x_i + \Delta x_i$ and $g_i = f_i + \Delta f_i$. Similar to the generalized Hough voting algorithm, the framework takes the ground truth bounding box center p_j and compute the Euclidean space offset $\Delta x_i \in \mathbb{R}^3$. The offset is explicitly supervised by a regression loss. The generated voting v_i will be aggregated around object center.

The next step is to generate vote clusters from $\{v_i = [y_i; g_i] \in \mathbb{R}^{3+C}\}$. Using farthest point sampling, a subset of the votes is sampled to get $\{v_{ik}\}$ with $k = 1, ..., K$. Then by finding all near votes withing a certain Euclidean distance, votes are aggregated into K clusters.

Finally, a two layer MLP is used to generate proposal and classification $p(C)$, which includes objectness score, bounding box parameter, scale parameters and semantic classification scores. Votes locations are transformed in to local normalized coordinate system by $z_i^{'} = (z_i - z_j)/r$ where z_j is the center location of the vote cluster. MLP_1 processes location and feature in each cluster independently and max-pooled in MLP_2 to share information in clusters. The loss function of the network includes an objectness

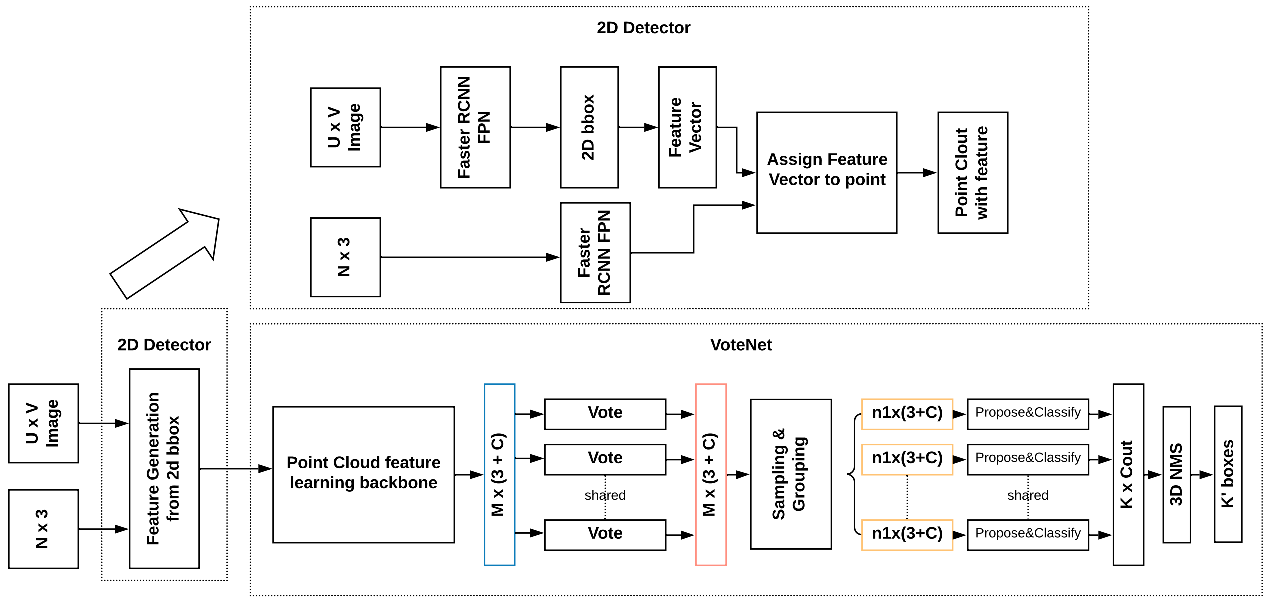


Figure 2: Model diagram of the proposed method. We take the point cloud and it's corresponding 2D image. If a 3D point is inside a 2D object frustum, we assign the 2D object score to the 3D point as dimention of feature.

loss, a bounding box estimation loss and a semantic classification loss. Furthermore, they are combined with voting loss into the total VoteNet loss function.

The framework is end-to-end and trained with an Adam optimizer. Diagram of the method is illustrated in the bottom part of Figure 2.

3.2. Embedding color information in high dimensional features

Recall that in the original VoteNet framework, points Color information can be an input as a feature of PointNet. Therefore the color channels can be input to compute enriched C-dimensional feature vector and thus embedded color in the feature vector. Then this feature vector will be used to train the MLP of Hough voting and generate proposal of vote clusters.

In fact, the VoteNet code on github already support this feature, by simply adding "use_color" to training and evaluation method. The author of the VoteNet did not include this in the original paper because the performance is very similar without using color. See section 4 and table 12 for details.

3.3. VoteNet with frustum hint

From the previous section we can see that VoteNet only use 3D point cloud location to detect 3D objects. In perfect situation 3D point cloud would have all the information of 2D image because 2D image is just a projection of the 3D points. However in reality, 2D image resolution is higher than 3D scans, thus potentially contains more information than 3D point clouds.

Inspired by Qi *et al.* [5] in their Frustum PointNet paper, we come up with the idea that frustum from 2D object detection could help 3D object detection by giving a hint. Specifically, we assume that in our dataset there are totally J categories. 2D object detector could generate 2d bounding box $[xmin, ymin, xmax, ymax]$ and class scores

$$b_{i,j} = [P(box_{(i,j)}|box_i \; belongs \; to \; category \; j)] \tag{1}$$

By projecting this bounding box back to 3D space, we get a frustum. If a point from point cloud $\{p_i\}_{i=1}^{N}$ is inside the frustum, then we assign the class score $b_{i,j}$ to it. We combine the score feature with it's 3D coordinate using

$$g_i = p_i, b_{i,j} \tag{2}$$

where p_i is a $1 * 3$ vector contains 3D location, and $b_{i,j}$ is a $1 * J$ vector that contains class scores of the bounding box. If a point doesn't belong to any frustum, simply make $b_{i,j}$ zero.

In our implementation, we do not explicitly calculate the frustum, but rather projecting point in point cloud into image space by using

$$(u, v) = M_{Rtilt} * M_K * (x, y, z) \tag{3}$$

	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
VoteNet	0.74	0.83	0.31	0.74	0.25	0.25	0.62	0.65	0.50	0.87	0.58
VoteNet use_color	0.69	0.83	0.33	0.74	0.25	0.27	0.59	0.65	0.49	0.90	0.57
Frustum PointNet	0.43	0.81	0.33	0.64	0.24	0.32	0.58	0.61	0.51	0.90	0.54
VoteNet+2D GT	0.84	0.95	0.71	0.82	0.63	0.82	0.87	0.89	0.74	0.95	0.82
VoteNet+2D	0.34	0.74	0.28	0.62	0.15	0.21	0.41	0.46	0.30	0.77	0.43

Table 1: Average precision on SUN RGB-D v1 dataset with 3D IoU threshold 0.25. **VoteNet** row contains results from original VoteNet paper, then we turn on ”use_color” and get **VoteNet use color**, which is described in section 3.2. We also put **Frustum PointNet** result here for reference. The method we proposed in section 3.3 is **VoteNet + 2D GT** and **VoteNet+2D**. The former one use ground truth bounding box as input while the latter one use RCNN output.

	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	AR
VoteNet	0.90	0.95	0.67	0.86	0.78	0.80	0.88	0.90	0.85	0.96	0.86
VoteNet use_color	0.84	0.94	0.67	0.85	0.79	0.76	0.88	0.90	0.84	0.97	0.84
VoteNet+2D GT	0.86	0.96	0.82	0.88	0.81	0.89	0.94	0.93	0.87	0.97	0.89
VoteNet+2D	0.48	0.83	0.58	0.74	0.62	0.57	0.64	0.71	0.73	0.86	0.68

Table 2: Recall rate on SUN RGB-D v1 dataset

Where M_{Rtilt} is the camera rotation matrix and M_K is translation and scaling matrix. After that we calculate if the projected point on the image is inside the 2D bounding box. The top part of Figure 2 shows the diagram of our proposed method.

4. Experiments

4.1. Implementation

Due to the fact that training of the VoteNet is computational heavy, the only dataset this project uses is SUN RGB-D v1[8]. The dataset contains 10335 captures and is divided into training set(5051-10335) and evaluation set(1-5050). Same with the VoteNet paper, We use a class whitelist that contains only 10 categories, as shown in table 1 and table 2.

For 2D object detection We use Detectron2[9]. It implements serveral state-of-the-art 2D detection and segmentation algorithms. We create a configuration based on faster_rcnn_R_101_FPN_3x. The SUN RGB-D dataset is transformed into MS COCO dataset format as input to Detectron2. Training the 2D detector takes 12 hours on a GTX 1060 6GB with 90000 iterations.

We also tried to implement and reproduce Frustum PointNet[5]. However the code released on Github does not include 2D detector for SUN RGB-D dataset. Due to the limitation of time We was not able to train Frustum PointNet. The results of this model in table 1 and table 2 are directly taken from the paper for reference.

We train the original VoteNet model as baseline performance. The training time for 180 epochs is 23 hours. Then We modify the model to accept bounding box information. Although the input tensor is larger, training time is roughly the same.

As a proof of concept, before using the real 2D object detector to generate bounding boxes, after training VoteNet W2D on training set, We use ground truth 2D bounding box on evaluation set as input. This is ”cheating” on the evaluation set because the frustum and it’s class are strong hints to the model. We would expect to see very high precision rate. In the next subsection the results are shown and discussed.

4.2. Results

Table 1 shows average precision and overall mAP of the models, and table 2 shows recall rate of the models. From the results we can see that using color as dimensions of feature does not improve the performance of the model. The reason could be, the scanned cloud points are so sparse so that information such as edges is not well preserved. Also PointNet is not optimized to extract color information in cloud points.

Before we use a real 2D detector, we tried a proof of concept method using ground truth 2D bounding box as input to the 3D detector. Results are shown as VoteNet + 2D GT in the tables. We can see mAP has been improved a lot. However if we look at Figure 3, on the first 3 rows, VoteNet + 2D GT results have more false positives. Because we are using the correct label as feature vector input, on the training set the model becomes ”lazy”. It will try to rely on the input feature directly rather than learn shape information from 3D location.

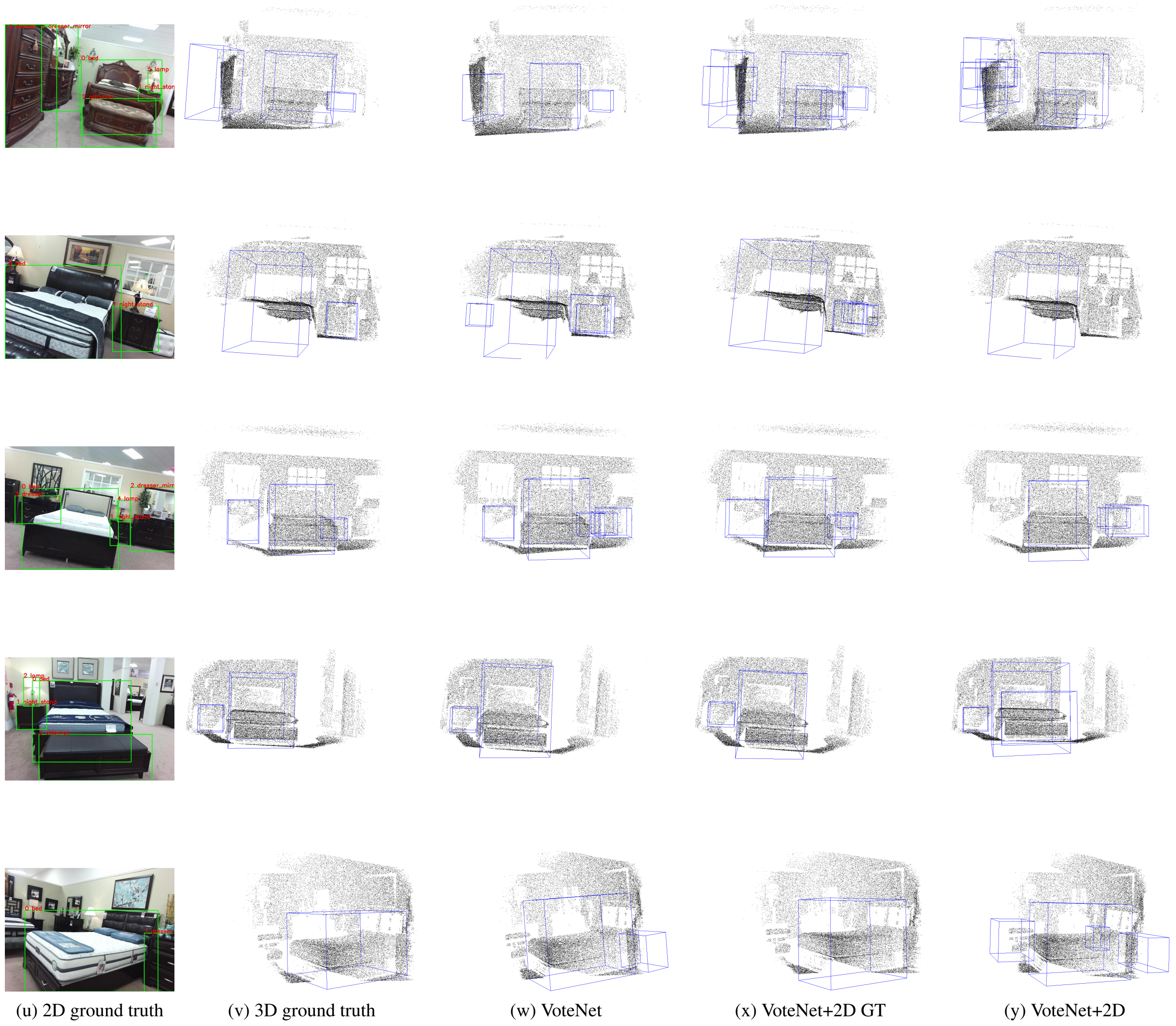

(u) 2D ground truth (v) 3D ground truth (w) VoteNet (x) VoteNet+2D GT (y) VoteNet+2D

Figure 3: Comparison of result from SUN RGB-D v1 dataset. The **first** column contains 2D bounding box ground truth from original dataset, and the **second** column contains 3D bounding box ground truth. Only classes on whitelist are shown in the image. The **third** column showes result from original VoteNet. The **fourth** column is from our proposed method using 2D ground truth, and finally the **fifth** column are produced from our method using RCNN 2D detector.

This issue become worse in our next experiment. We use a faster RCNN FPN to generate 2D bounding box on evaluation set. Results are shown as VoteNet + 2D. The precision and recall rate have dropped a lot. Moreover, we find that the trust threshold when converting 2D object detection result to 3D point feature has a great impact to mAP. We tried different values and find 0.4 gives the best mAP. On the fifth column of figure 3 we can still see a lot false positive results. Our model might be overfit to the 2D feature. Although our proposed method doesn't do well in this evaluation, we think it could be improved. We will discuss this in the next section.

5. Conclusion

We proposed a method that utilizes 2D RGB information trying to improve 3D point cloud object detection performance. In our proof of concept we see some improvements but when we use a real 2D detector the results are getting worse.

We think the model has a lot of potential for improvement. We could use more precised 2D detector in the first part, and choose trust threshold more carefully. More over, we could divide training set into 2D training set and 3D training set, and use non-perfect 2D bounding box to train 3D model. This could help to prevent model overfitting. Finally the model is not end-to-end, and it would be more efficient if we can transform it into one.

The code of this project can be found at https://github.com/xdong086/cs229project

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[2] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018.

[3] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *In ECCV workshop on statistical learning in computer vision*, pages 17–32, 2004.

[4] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[5] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. 2016. cite arxiv:1612.00593Comment: CVPR 2017.

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 2017. cite arxiv:1706.02413.

[8] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576. IEEE Computer Society, 2015.

[9] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[10] Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey, 2019.